Application Note

BCM63XX/BCM68XX

NAND Flash Support

CPE-AN1102-R

5300 California Avenue • Irvine, CA 92617 • Phone: 949-926-5000 • Fax: 949-926-5203

January 5, 2016

Revision History

Revision	*Date*	*Change Description*
CPE-AN1102-R	05/01/16	**Updated:** • "NAND Flash Update Procedure" on page 7 • "NAND Flash Partitions" on page 9 • Table 1: "NAND Flash Partitions," on page 10 • "NAND Flash Parts" on page 12 • "Spare Area Format" on page 13 • "NAND Flash Source Files" on page 17 • "SPI NAND Flash Source Files" on page 17 **Added:** • Added support for BCM6848, BCM6858, BCM4908 • "Image Partitions" on page 11 • "Manually Changing Partition Sizes" on page 12 **Removed:** • Supported NAND Flash Devices table • Supported SPI NAND Flash Devices table
CPE-AN1101-R	02/23/15	**Updated:** • "Preparing the Image" • Figure 1: "NAND Flash Partitions" • "NAND Flash Parts" • Table 2: "Useful Options for the NAND Image Build Utility" **Added:** • Table 2: "Supported SPI NAND Flash Devices"
CPE-AN1100-R	03/23/14	Initial release

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Table of Contents

List of Figures

List of Tables

About This Document

Purpose and Audience

NAND Flash is supported on the Broadcom® BCM63XX, BCM63XXX, BCM68XX, and BCM68XXX chips. This document provides information about NAND Flash support on these chips. It is meant for hardware and software engineers working with the chips and their reference design boards.

Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Broadcom documents, go to: http://www.broadcom.com/press/glossary.php.

Document Conventions

The following conventions may be used in this document:

Convention	*Description*
Bold	User input and actions: for example, type **exit**, click **OK**, press **Alt+C**
`Monospace`	Code: `#include <iostream>` HTML: `<td rowspan = 3>` Command line commands and parameters: `wl [-l] <command>`
< >	Placeholders for *required* elements: enter your <username> or `wl <command>`
[ ]	Indicates *optional* command-line parameters: `wl [-l]` Indicates bit and byte ranges (inclusive): [0:3] or [7:0]
directory/*	Refers to everything in the directory

Technical Support

Broadcom provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates through its customer support portal (https://support.broadcom.com). For a CSP account, contact your Sales or Engineering support representative.

In addition, Broadcom provides other product support through its Downloads and Support site (http://www.broadcom.com/support/).

Overview

NAND Flash is supported for the following Broadcom devices:

BCM6362	BCM63381	BCM6838	BCM4908
BCM6328	BCM63138	BCM6848	
BCM63268	BCM63148	BCM6858	

This Application Note provides the following information about the NAND Flash support.

- NAND Flash Update Procedure
- NAND Flash Partitions
- NAND Flash Dual Image Support
- NAND Flash Parts
- NAND Image for NAND Programmer
- NAND Flash Source Files

To use NAND Flash, the reference design board must be strapped to boot from NAND Flash. Only one NAND Flash chip may be used. Reference platform software does not support NAND Flash and NOR Flash simultaneously; but if necessary, this can be achieved using source code modifications.

While a pure JFFS2 filesystem is supported, it exists essentially for legacy purposes and the split UBI filesystem should always be used in preference to the JFFS2 version.

NAND Flash Update Procedure

Setting Default Partition Sizes

The software allows up to four data partitions. By default, the 4th data partition will always be allocated with 4MB. 1, 2, and 3 are initialized to 0MB.

To override these defaults, change the PROFILE. When the software is built, it will create a default partition table based on the new values.

```
BRCM_MISC1_PARTITION_SIZE=0
BRCM_MISC2_PARTITION_SIZE=0
BRCM_MISC3_PARTITION_SIZE=0
BRCM_MISC4_PARTITION_SIZE=4
```

The data partition sizes can be changed by the CFERAM using b or x command. e.g.

```
CFE>     x
Press: <enter> to use current value
        '-' to go previous parameter
        '.' to clear the current value
        'x' to exit this command
Partition 1 Size (MB)            :  0M
```

```
Partition 2 Size (MB)             :  0M
Partition 3 Size (MB)             :  0M
Partition 4 Size (MB) (Data)      :  4M
```

Typically, when data partition sizes are changed, it's advisable to reprogram the binaries.

Preparing the Image

The following steps write a Linux router image to an uninitialized SPI or parallel NAND Flash part. The BCM63138 is used as an example. Most examples also use TFTP for file transfer, but the HTTP server is usually preferable.

1. On the Linux build PC, run **make menuconfig** from the base directory.
   ```
   $ cd SOURCE_DIR
   $ make menuconfig
   ```
2. In the menuconfig application, load the desired profile, visit the Root Filesystem submenu, and ensure that the block size for your flash is among the options selected. (See "Setting Default Partition Sizes" on page 7.)
3. On the Linux build PC, build the desired profile.
   ```
   $ make PROFILE=96313GW
   ```
   Or
   `$ make PROFILE=96313GW nandcfeimage` to produce a minimal CFE-only NAND image that is much faster to load.

JTAG Debugger

1. On the Linux build PC, copy the bcm963138GW_cferom_fs_flash_image_*.w image to a TFTP boot directory.
   ```
   $ cp targets/963138GW/bcm963138GW_nand_cferom_fs_image_*.w /tftpboot/.
   ```
2. Set the board strap for NAND boot with proper ECC setting and page size according to the NAND chip on board. The ECC must be at least as strong as required by the Flash device. For SoCs that support BCH-4, *BCH4* should always be used in preference to Hamming even when the Flash requires only 1-bit correction.
3. On the Linux build PC, build the BCM63138 CFE bootloader.
   ```
   $ cd SOURCE_DIR/cfe/build/broadcom/bcm63xx_rom
   $ make BRCM_CHIP=63138 BLD_NAND=1
   ```
4. Launch the JTAG debugger.
5. From the JTAG debugger, connect to the CPU and run any needed memory initialization script, then load the CFE RAM ELF file.
   The CFE RAM ELF file is located at SOURCE_DIR/cfe/build/broadcom/bcm63xx_ram/cfe63138.
6. From the JTAG debugger, start the CFE RAM ELF file. The CFE bootloader is started on the serial console. Configure the NVRAM parameters as required.
7. To download and flash the NAND Flash image, use the following command from the BCM63138 router serial console:
   - For 128 KB block size NAND Flash chips: `w bcm963138GW_nand_cferom_fs_image_128_ubi.w`

When the image is flashed, the BCM63138 router will boot the Linux image that is stored on NAND Flash. The image with `cferom` in the name starts to flash the image at block 0. The (upgrade) image without `cferom` in the name starts to flash the image at block 1.

Subsequent image updates can be done from either the CFE bootloader or a Linux router.

To load from a TFTP server, enter the *c* command from the BCM63138 router serial console, to configure the Host IP address of the Linux PC where the TFTP server and the NAND Flash image are located.

To load from a web browser, configure a PC with a static IP address on the 192.168.1.0/24 subnet, navigate to http://192.168.1.1, and upload the image file via the form provided.

SPI Flash

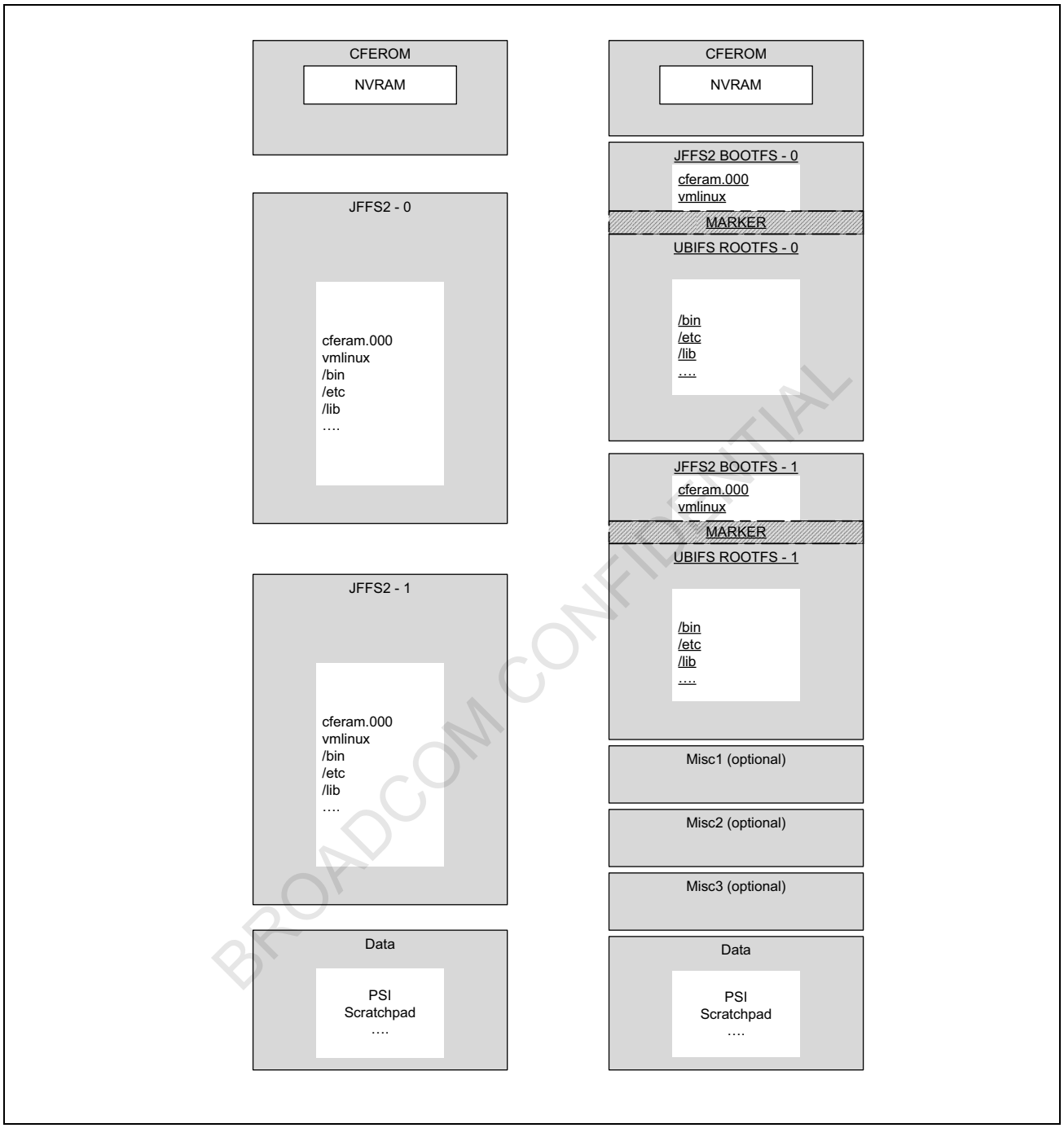
The following steps are for customers who do not have a JTAG debugger. The procedure supports CPE boots from SPI Flash and upgrades the image to NAND Flash. The board BCM963138REF is used as an example.

1. On the Linux build PC, build 963138 bootloader to support the program image to NAND Flash.
```
$ cd SOURCE_DIR/cfe/build/broadcom/bcm63xx_rom
$ make BRCM_CHIP=63138 BLD_SPI_NAND=1
```
2. Change the bootstrap to boot from SPI Flash, but leave ECC switches set correctly as if booting from NAND.
3. Upgrade `SOURCE_DIR/cfe/build/broadcom/bcm63xx_rom/bcm963138_cfe.w` to the board.
4. When the image upgrade is complete, change the bootstrap to boot from NAND Flash.
5. Boot and adjust NVRAM settings as required.

NAND Flash Partitions

Figure 1 shows the JFFS2 and UBIFS NAND Flash partitioning.

Figure 1: NAND Flash Partitions



Raw CFE ROM Boot Partition

The first NAND Flash block contains a raw partition with the CFE ROM code. It executes in place. It determines which JFFS2 file system partition to boot from by reading the boot flag from the NVRAM configuration. It then loads the CFE bootloader RAM image into memory from the JFFS2 file system partition and jumps to its entry point. In the case of BCM68500 devices, CFE_ROM is replicated eight times in erase blocks 0 to 7, and the first good copy is used automatically.

Image Partitions

There are two image system partitions that contain the files used in a Linux router image.

JFFS2 Images

JFFS2 images contain a single read-only filesystem that is mounted as the root filesystem. Two of the files are the CFE RAM image and the Linux kernel, vmlinux. The file extension on the CFE RAM image file is a number between 000 and 999. The partition with the CFERAM file bearing the higher number is the most recently flashed partition.

UBI Images

UBI images are actually "split" images. They contain a small read-only JFFS2 filesystem, known as the "boot" filesystem followed by a marker, one erase-block in size, followed by a root filesystem consisting of a UBIFS in a UBI volume.

MISC1, MISC2, and MISC3 Partitions

If the sizes of MISC1, MISC2, and MIS3 partitions in the NVRAM are non-zero, space will be reserved between the end of the image area and the data partition. These partitions are generally ignored by CFE, though they can be imaged by CFE and Linux will create MTD devices for them.

Attaching them or mounting them is a function of startup scripts that can be customized.

JFFS2 Data Partition

The read-write JFFS2 data partition contains files that require write access such as configuration data (psi) and data that is needed across reboots (scratch pad).

Raw Bad Block Table Partition

The bad block table partition maintains a list of NAND blocks that are bad.

NAND Flash Dual Image Support

The file system part of a NAND Flash image is flashed to the nonactive file system partition. The file system partition to boot from is read from the NVRAM configuration boot flag. The value of this field is *boot from latest image* or *boot from previous image*. This field is configured in the CFE serial console *c* command.

When an image is flashed, the extension of the JFFS2 file, cferam.xxx, is changed to a number that is greater than the highest number that is currently on the two file systems. This file name is used to identify the latest image.

Manually Changing Partition Sizes

Normally, in manufacturing, the desired MISC partition sizes are programmed in the NVRAM in advance and the partition table is loaded on first boot.

When repartitioning, only image 0 and cferom are preserved.

To change partition sizes, upgrade **twice** to guarantee that the current version is available on image 0, then use the "b" command to update parameters and select the appropriate sizes. After reboot the new partitions will be available.

NAND Flash Parts

In order to boot from NAND Flash, the chip's NAND Flash controller must be able to determine the Flash part's configuration. This is accomplished by retrieving the configuration from the ONFI parameter table if the NAND device supports ONFI. If not, the NAND Flash controller uses an internal list of NAND Flash chip identifiers and corresponding configurations. If the NAND Flash chip is not in the list, the NAND Flash controller uses the fourth byte of the chip ID as the page size and tries to determine the configuration. This means that a NAND Flash part that is not in the list may work, but must first be verified.

For a parallel NAND part to work with the Broadcom DSL chips:

- The DSL chip must be able to support the ECC level required by the NAND, and the NAND must have enough OOB/spare area bytes to support the ECC level required.
- The NAND MUST NOT do its own ECC, i.e., the DSL chip must be able to determine exactly how many bits are bad in a page.
- NAND chips with number of operations (NOP) >= 4 will work. (NOP < 4 may work, but in this case they must be identified properly by the system).
- NAND Flash parts must be "CE Don't Care".
- NAND Flash parts must have guaranteed good block 0 (zero).
- NAND Flash size must be at least 128 MB.

NAND Image for NAND Programmer

Customers may need to build an image with spare area data if they use a NAND programmer to load the image in the product line. Broadcom provides a utility to build such an image.

Spare Area Format

The spare area contains the JFFS2 clean marker, bad block indicator (BI), and Error Correction Code (ECC). The JFFS2 clean marker is 8 bytes of fixed pattern 0x19,0x85,0x20,0x03,0x00,0x00,0x00,0x08. BI is 0xFF if good, and non-0xFF if bad. The ECC calculation may use the Hamming Code, BCH-4, BCH-8, or BCH-12 algorithms, depending on the chip and its configuration. The ECC must be at least as strong as required by the Flash device. For SoCs that support BCH-4, *BCH4* should always be used in preference to Hamming even when the flash requires only 1-bit correction.

Each character in the following layouts represents a byte in the spare area. The key is:

- B = location of bad block indicator
- E = location of ECC
- 0 = location for spare area data, the first eight locations contain the JFFS2 cleanmarker.

Table 1: Spare Area Formats

Spare Size	*ECC Algorithm*	*Layout*
64 bytes	Hamming	B,B,0,0,0,0,E,E,E,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,0,0,0,0,0,0,0,0 0,0,0,0,0,0,E,E,E,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,0,0,0,0,0,0,0,0
6 bytes	Hamming	0,0,0,0,0,B,E,E,E,0,0,0,0,0,0,0
16 bytes	BCH-4	0,0,0,0,0,B,0,0,0,E,E,E,E,E,E,E
128 bytes	Hamming	B,B,0,0,0,0,E,E,E,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,0,0,0,0,0,0,0,0 0,0,0,0,0,0,E,E,E,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,0,0,0,0,0,0,0,0 0,0,0,0,0,0,E,E,E,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,0,0,0,0,0,0,0,0 0,0,0,0,0,0,E,E,E,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,0,0,0,0,0,0,0,0
128 bytes	BCH-4	B,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E
64 bytes	BCH-4	B,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E
64 bytes	BCH-8	B,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E
128 bytes	BCH-8	B,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E

Table 1: Spare Area Formats (Cont.)

Spare Size	***ECC Algorithm***	***Layout***
216 bytes	BCH-8	B,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E
216 bytes	BCH-12	B,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E
256 bytes	BCH-4	B,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,9,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E
256 bytes	BCH-8	B,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,9,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E

Table 1: Spare Area Formats (Cont.)

Spare Size	***ECC Algorithm***	***Layout***
432 bytes	BCH-8	B,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E
432 bytes	BCH-12	B,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E 0,0,0,0,0,0,0,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E,E

Example: In the system using 16 bytes spare area and Hamming ECC algorithm, a dump of the spare area using the *e s* command in CFE shows:

0x19,0x85,0x20,0x03,0x00,0xff,0x3c,0x0f – 0x00,0x00,0x00,0x08,0xff,0xff,0xff,0xff

NAND Image Build Utility

The spare area data is not contained in the image generated by the make PROFILE=XXX. A utility, *createnfimg*, can be run that calculates the spare area and creates an image that contains both the page data and the spare area data.

The tool is located in the hostTools/createnfimg folder from the release source. Contact Broadcom support if it is not available. Run the following command to build the tool:

```
gcc -o createnfimg createnfimg.c
```

This program takes in a binary file and breaks it into 512-byte or 2048-byte pages. For each 512-byte partial page, it creates the NAND OOB area contents with a Hamming or BCH ECC and optional JFFS2 cleanmarker. Each page is written out, followed by the OOB area, which is 16 bytes for a 512-byte page or 64 bytes for a 2048-byte page. The output file name is the input file name with a `.out` extension.

Table 2: Useful Options for the NAND Image Build Utility

Option	*Description*
-b <bch_lvl>	Level of ECC correction bit. Default = 4. • 1 for 1-bit Hamming • 4 for 4-bit BCH • 8 for 8-bit BCH
-m <field_order>	Order of the finite field for BCH[a]. Default = 13.
-c <partial_page_size>	Partial page size in bytes. Default = 512.
-i <infile>	Name of the input file. Output files will be `infile.out`.
-p <page_size>	Page size in bytes. Default = 2048.
-l <0|1>	Set little-endian clean marker for ARM based chips (BCM68138, BCM68148).
-d <pad_amount>	OOB padding amount to fill after ECC is finished, for example with newer NAND devices that have 128 bytes OOB per page, treat part like OOB=64 (-r 16) and set this value to 64 to pad 64 bytes after the OOB 64 bytes. Default = 0.
-r <oob_size>	OOB size in bytes per 512 byte subpage. Default = 16
-j <0|1r>	1 = Add JFFS2 cleanmarker to spare area. Default = 1
-n <pages_per_block>	Pages per block. Default = 64.
-t <0|1>	Trim input file to page boundary (removes extraneous signature bytes). Default = 1.

a. The BCH finite field order, the *m* parameter:
– is 13 for the BCM6368, BCM6816, BCM6362, BCM6328, and BCM63268.
– is 14 for the BCM6828, BCM6838, BCM6848, BCM6858, BCM6818, BCM63138, BCM63148, BCM4908 and newer chips.

Usage examples:

BCM63268

- Large page BCH-8:

```
./createnfimg -b 8 -P 2048 -m 13 -i bcm963268GW_nand_cferom_fs_image_128_ubi.w
```

BCM63138

- Large page BCH-8:

```
./createnfimg -b 8 -P 2048 -m 14 -i bcm963138GW_nand_cferom_fs_image_128_ubi.w
```

NAND Flash Source Files

The following files have changes for NAND Flash support.

- hostTools/scripts/defconfig-bcm.template
- hostTools/scripts/gendefconfig
  These files contain kernel configuration information to configure NAND Flash MTD driver when JFFS2 is configured in make menuconfig.
- hostTools/jffs2/*
  Source files for the mkfs.jffs2 utility, which creates the JFFS2 file system used with NAND Flash images.
- hostTools/createnfimg/*
  Source files for the NAND image build utility.
- cfe/cfe/board/bcm63xx_rom/src/bcm63xx_main.c
  This file contains bootup code that reads the CFE RAM image from JFFS2.
- cfe/cfe/board/bcm63xx_ram/src/*
  These files contain NAND Flash support for booting Linux and flash image update.
- targets/buildFS2
  This script creates the JFFS2 file system.
- shared/opensource/flash/nandflash.c
  NAND Flash driver that is used by the CFE bootloader.
- kernel/linux/drivers/mtd/brcmnand/*
  NAND Flash MTD driver that is used by the Linux router.

SPI NAND Flash Source Files

SPI NAND is an extension of NAND, and in addition uses the Linux NAND infrastructure. Most of the files used for NAND are also used for SPI NAND, with the exception of noted replacements. The following files have changes for SPI NAND Flash support:

- shared/opensource/flash/nandflash.c → shared/opensource/flash/spinandflash.c
  SPI NAND Flash driver that is used by the CFE bootloader.
- kernel/linux/drivers/mtd/brcmnand/* → kernel/linux/drivers/mtd/nand/bcm63xx_spinand.c & nand_ids.c, kernel/linux/drivers/mtd/maps/bcm963xx_mtd.c
  SPI NAND Flash MTD driver that is used by the Linux router.

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Phone: 949-926-5000
Fax: 949-926-5203
E-mail: info@broadcom.com
Web: www.broadcom.com